In *Proceedings of the 17th Annual Computer Security Applications Conference*, pages 240–251, New Orleans, Louisiana, Dec. 10–14, 2001, published by the IEEE Computer Society.

eXpert-BSM: A Host-based Intrusion Detection Solution for Sun Solaris[*]

Ulf Lindqvist and Phillip A. Porras
System Design Laboratory
SRI International
333 Ravenswood Avenue
Menlo Park, CA 94025-3493
{ulf, porras}@sdl.sri.com

Abstract

eXpert-BSM *is a real time forward-reasoning expert system that analyzes Sun Solaris audit trails. Based on many years of intrusion detection research,* eXpert-BSM*'s knowledge base detects a wide range of specific and general forms of misuse, provides detailed reports and recommendations to the system operator, and has a low false-alarm rate. Host-based intrusion detection offers the ability to detect misuse and subversion through the direct monitoring of processes inside the host, providing an important complement to network-based surveillance. Suites of* eXpert-BSM*s may be deployed throughout a network, and their alarms managed, correlated, and acted on by remote or local subscribing security services, thus helping to address issues of decentralized management. Inside the host,* eXpert-BSM *is intended to operate as a true* security daemon *for host systems, consuming few CPU cycles and very little memory and secondary storage.* eXpert-BSM *has been available for download on the Internet since April 2000, and has been successfully deployed in several production environments.*

1. Introduction

When research on intrusion detection was initiated in the early 1980s, the problem was often referred to as automated audit-trail analysis. In theory, auditing is an important security service that both establishes accountability for users and aids in damage assessment once an abuse is discovered. Unfortunately, in practice the volumes of data that tend to be produced by audit services are such that any security violation recorded within the audit trail is often secure from discovery as well. The increasing speed and complexity of modern computing environments has increased the volumes of audit data that can be produced.

The Solaris Basic Security Module (BSM) [21] is one example of an auditing facility that can provide detailed records about system events. However, for system operators lacking intelligent analysis tools, there are two dominant strategies that emerge in using the audit facility:

1. Turn on auditing for all or most event types, and have a careful scheme in place for copying the large amounts of audit data to secondary storage for its potential use later in forensic analysis.

2. Do not perform auditing at all.

Neither approach utilizes the full potential of auditing facilities as an important contributor to a system's operational security.

The EMERALD (Event Monitoring Enabling Responses to Anomalous Live Disturbances) environment is a distributed scalable tool suite for tracking malicious activity through and across large networks [17]. EMERALD introduces a highly distributed, building-block approach to network surveillance, attack isolation, and automated response. A central concept of EMERALD is its distributed, lightweight *monitors*, diverse with respect both to the monitored event streams and to analysis techniques. *eXpert-BSM* represents one example of an EMERALD monitor that can stand alone as an important host protection service, and can also be easily configured to fit into a distributed framework of surveillance, correlation, and response.

[*]The work presented in this paper was funded by the Defense Advanced Research Projects Agency (DARPA), under contract number F30602-98-C-0059. SRI International, EMERALD, and eXpert-BSM are trademarks of SRI International. Sun, Sun Microsystems, Solaris, SunOS, Sun Enterprise and SunSHIELD are trademarks or registered trademarks of Sun Microsystems, Inc. in the United States and other countries. SRI International is independent of Sun Microsystems, Inc. All other registered trademarks, trademarks, trade names or service marks are the property of their respective owners.

eXpert-BSM is a security service for isolating misuse and other security-relevant warning indicators from the Sun Solaris audit facility. Initial development of *eXpert-BSM* began in 1998 and has continued to present. This paper is the first publication describing the design and features of *eXpert-BSM* and how it fills a vital function in security coverage not provided by network intrusion detection services. Section 2 discusses the complementary nature of host audit trail analysis and network traffic monitoring. Section 3 summarizes the *eXpert-BSM* attack coverage. Section 4 presents the *eXpert-BSM* capabilities and unique features while Section 5 discusses deployment experiences and performance characteristics. Section 6 discusses related work in the area of host-based security analysis.

2. Audit data vs network traffic

An intrusion detection system (IDS) analyzes an event stream in an attempt to categorize the events as normal or intrusive. The first IDSs proposed and developed in the early 1980s were host based, analyzing the audit trails of mainframe computers in search of anomalies and signs of malicious activity. When later applied to networked environments, the dominant architecture was centralized collection and analysis of raw audit data from multiple hosts. The first network-based IDS, using data “sniffed” from a broadcast Ethernet network, was NSM from UC Davis [8]. The network-based trend that followed has been so strong in commercial and free IDSs that many people equate intrusion detection with network traffic analysis.

In this paper, we somewhat narrowly use the term *host-based* to refer to a monitor that analyzes audit data from the operating system kernel. In referring to host-based intrusion detection, others have included any form of analysis that is focused on the protection of a single host. For example, some IDS developers have proposed placing a network event collector and analyzer locally on every host, observing traffic involving only that host. That would not fit into the definition of host-based analysis as used in this paper. Accordingly, *network-based* analyses are defined here to involve the analysis of network traffic data, wherever the monitor is located.

Major functional separation between host versus network-based analyses arises from the content of the data streams being analyzed. Audit-based analyses provide an exceptional degree of insight into the internal operations of processes executing within the host. From the audit-trail vantage point, one can examine all access control decisions occurring between the kernel and user processes, profile normality in process activity, and compare user actions against their expected roles within the system.

Surveillance through network traffic analysis allows a system to view the network communications across multiple hosts. In broadcast networks, a single sensor can provide analysis coverage over an entire local area network (LAN). Both host- and network-based surveillance are important and complementary. Each has its place in the arsenal of INFOSEC devices being made available to supplement the need for computer and network security. However, each approach has its respective weaknesses.

2.1. Network-based IDS limitations

A fundamental limitation to network analysis is that not all forms of misuse will necessarily generate network traffic. Further, not all misuse activity that results in network traffic will provide sufficient information to isolate the misuse. Examples of such information include the true full local pathname of a file retrieved through HTTP, or the user ID under which a particular service daemon executes. This is also a problem with buffer overflows and other well-known malicious attacks that are performed from the console or over an encrypted channel.

Application-layer encryption of network traffic is becoming more common and user transparent thanks to technology such as SSL-enabled Web browsers and Secure Shell (ssh). The same is true for lower-layer encryption through virtual private networks, some of which are based on the IPSEC standard. While this is a positive step forward in communications integrity and the prevention of data theft, it makes network-based intrusion detection more difficult as potentially malicious instructions are also encrypted.

Another problem with network intrusion detection involves the evolution of common network topologies, specifically, the growing popularity of non-broadcast networks. Inserting a network sniffer in the path of all LAN traffic is becoming more challenging. For example, switching technology allows improved network performance by effectively turning a broadcast Ethernet network into a unicast network, hampering sniffing opportunities. Also, if there are multiple possible routes between two communicating hosts, some packets could be routed around the sniffer location.

When intercepting and analyzing the communication between two hosts, it is of paramount importance for correct analysis that the traffic is interpreted equally by the IDS and the receiver. If not, the IDS could be tricked into interpreting traffic as benign while the receiver, making a different interpretation, becomes the victim of an attack. With respect to IP stacks, there are many subtle differences among operating systems that could be used by an attacker to send instructions that appear benign to the IDS, but have malign effects on the victim host [19]. The same holds for application-level interpretation. For instance, Web servers for the Windows platform tend to accept the backslash character as a valid path separator in addition to the forward slash, while servers on Unix platforms do not.

Network traffic analysis is also challenged with the need to provide transaction and session reconstruction, requiring great efficiency in managing state. In many cases, a single packet is not sufficient to correctly identify intrusive behavior. For advanced analysis, the IDS must reconstruct transactions and sessions based on the observed data and therefore keep potentially large amounts of state information for arbitrarily long periods of time. Merely combining the requests and replies across many parallel sessions into transactions can be a complex task for the IDS.

Finally, there is the issue of scalability of a network IDS to large traffic volumes. For line speeds where relatively simple routing decisions have to be made in firmware to be sufficiently fast, the more complicated analysis required by an IDS implemented in software has little chance to keep up.

2.2. Host-based IDS limitations

Host-based intrusion detection can avoid most, if not all, of the problems listed above. Thus, it is an important complement to the threat coverage of network-based monitoring. However, host-based monitors also have a set of general problems associated with them.

As with network traffic analysis, host-based analysis is limited by the available content in the event stream. For example, a host-based monitor can fail to observe network-related activity. This illustrates the complementary nature that host analysis shares with network traffic analysis tools. Unfortunately, the use of network-based vulnerability scanners has become a prominent practice in security evaluation procedures, and an evaluator pointing a network scanner against a host equipped with a host-based IDS is often disappointed when the IDS does not react to all elements of the scan. Very severe host attacks readily detectable with host-based analysis are similarly often not recognizable by network IDSs.

Another potential issue with a host-based IDS is its vulnerability to attack once a system has been compromised. When an attacker has taken over the omnipotent super-user account (root, administrator), then, in the absence of automated response, the IDS is itself subject to attack. If the IDS transmits alarm information over the network to another entity, it may be able to report super-user subversion to others before the attacker can stop the IDS.

If a denial-of-service attack brings down the host, the IDS will go down with it. The IDS may be able to raise an alarm about a resource-exhaustion attack in progress, while there could be other attacks that crash the host with only a minimal number of network packets, before the IDS can send out an alarm. The additional load put on the host by the IDS monitor could also be of concern.

3. eXpert-BSM knowledge base

Among the first steps toward developing an effective and maintainable misuse detection service is to select a reasoning strategy and knowledge representation structure that is well suited and efficient for this problem domain. In [10], we argue why forward-chaining rule-based systems are highly useful for computer and network intrusion detection. The core of *eXpert-BSM* consists of an inference engine and knowledge base built with the Production-Based Expert System Toolset (P-BEST), a highly optimized forward-chaining rule-based system builder for real-time event analysis.

In the field of expert system analysis, forward-chaining strategies dominate applications that provide prognosis, monitoring, and system control. Generally, forward-reasoning systems excel in expressing logical inferences across multiple events in search of specific event sequences or activity that crosses predefined thresholds of normalcy. *eXpert-BSM*'s P-BEST models can comprehend intrusive behavior that may involve complex/multiple event orderings with elaborate pre- or post-conditions. This allows for a concise rule base, while still being able to recognize wide variation in intrusive activity.

In contrast, a variety of signature-based intrusion detection techniques employ stateless reasoning to isolate single-step malicious activity, such as rudimentary pattern matching. For very high-volume event analysis, stateless predicate reasoning can be quite effective for simple single-packet exploit detection. However, limited expressibility in misuse definitions can lead to inflated rule bases to cover all variations of a known phenomena. Rudimentary pattern matching also fails to cover multi-event scenarios.

From 1996 to present, P-BEST has been employed in the development of nine independent intrusion-detection engines under the EMERALD framework of distributed sensors managed under a correlation hierarchy. P-BEST has shown itself to be an effective real-time transaction processing system, with a pre-compilation library that allows its inference engines and knowledge bases to be easily integrated into large program frameworks. Its language is small and easily extendible, as calls to arbitrary C functions are possible anywhere in the rule structure. Since its inception on Unix, P-BEST has undergone many optimizations (Section 5.3).

The P-BEST toolset consists of a rule translator and a library of run-time routines. When using P-BEST, rules and facts are written in the P-BEST production rule specification language. The misuse detection P-BEST compiler, *pbcc*, is then used to translate the P-BEST knowledge specification into a callable expert system library. A full discussion of the P-BEST language definition with examples is provided in [10].

3.1. eXpert-BSM attack coverage

The *eXpert-BSM* knowledge base consists of 123 P-BEST rules, which allow *eXpert-BSM* to recognize 46 general forms of misuse or warning indicators of abuse. Initial development of this rule base began in 1998 and has continued to present. Based on experimental evaluations (see Section 5.2) and other input, *eXpert-BSM*'s knowledge base has been refined and extended into an effective suite of intrusion models for identifying, where possible, the broadest forms of activity that indicate abusive or intrusive activity on Unix hosts.

eXpert-BSM excels at detecting when an adversary attempts to violate security on the host, regardless of whether it is an external agent or an insider operating from the console. While attack space coverage is not claimed to be complete, significant effort and experience has been invested in this knowledge base over several years. The majority of the intrusion models attempt to recognize the most general form of misuse by detecting state transitions that represent the underlying compromise of intrusions and other known misuse activity. With respect to attack coverage, these intrusion models are categorized under the following broad areas of host misuse:

Data Theft — involves attempts by non-administrative users to perform read operations on files or devices in a manner considered inconsistent with the system security policy. This includes attempts to access files stored in nonpublic directories that are owned by other users, or to reference files in violation of *eXpert-BSM*'s surveillance policy (Section 3.2). The category includes attempts to access root core file contents, a well-known method to gain access to encrypted or even cleartext password content. The category also includes opening network interface devices in promiscuous mode, indicating attempts by non-administrative users to sniff traffic from the network.

System/User Data Manipulation — attempts by non-administrative users to modify system or user data, where *modify* broadly means attempts to alter, create, overwrite, append, remove, or change content or the attributes of file system objects. Coverage in this category includes attempts to modify system files within which security-relevant configuration parameters are stored. This configurable list of files typically includes files in */etc*. The category also includes attempts by anonymous FTP users to modify file system content outside a predefined upload directory, should one exist. Intrusion models that detect attempts to modify user environment files (e.g., *.csrhc*, *.login*, or *.rhosts*) or modify files in violation of *eXpert-BSM*'s surveillance policy (Section 3.2) are also included in this category. Lastly, the category provides comprehensive coverage over attempts to modify system executable binaries and scripts stored in publicly shared binary directories.

Privilege Subversion — provides broad and effective coverage of illegal attempts to subvert root or other administrator authority, either through the illegal changing of one's operating authority, subverting the function of a privileged (setuid) application, or by causing a setuid process to execute an application that is not owned by the setuid program owner or the system. Intrusion models in this category are capable of detecting the three variations of buffer overflow attacks [14, 2] that continue to plague Unix setuid applications and inetd services: exec argument buffer overflows, environment variable overflows, and data-segment overflows. The generality of *eXpert-BSM*'s buffer overflow model was demonstrated when a new such exploit was published for Solaris 8 [5]. Without being updated with specific knowledge about the new attack, *eXpert-BSM* detected and correctly identified this event as privilege subversion through a buffer overflow. Overall, the BSM audit trail provides far superior event content than network traffic from which to identify process subversion on hosts. Inetd service subversion (such as the well-known *sadmind(1)* attack on Solaris [3]) is an example of a data segment buffer overflow, while exec argument and environment variable overflows have been exploited in perhaps more than a dozen setuid applications on Solaris alone (e.g., *eject*, *fdformat*, *ffbconfig*, *passwd*, *ping*, *rdist*, *rlogin*, *ufsrestore*, and *xlock*).

Account Probing and Guessing — identifies repeated attempts to enter a system via authentication services such as rlogin or FTP, or attempts to gain root authority by non-administrative users or from external clients.

Suspicious Network Activity — recognizes various attempts to probe or scan the host, or misuse the host's FTP services to distribute content to other external sites (i.e., FTP warez hosting). While BSM audit trails provide minimal insight into raw network traffic activity, they do allow the monitor to recognize successful connections to TCP-based services, and more importantly provide detailed insight into the internal operations of the network server process as it is being used (or potentially misused). *eXpert-BSM* also provides a TCP port scan detection capability on TCP ports that have enabled services on the host.

Asset Distress — identifies operational activity that indicates a current or impending failure or significant degradation of a system asset. The majority of these problems are very difficult, if not impossible, to diagnose via network traffic analysis. This category includes filesystem or process table exhaustion, and also core-dump events by root-owned services. In addition, this category includes detection of malicious service denials, both through remote agents attempting to exhaust process tables via inetd services, and by a Solaris-specific self echo flooding attack by local host processes.

User-specifiable Surveillance — allows the *eXpert-BSM* operator to create site-specific policies on activity that should trigger an immediate alarm. This category includes the ability to recognize operator-defined command arguments considered suspicious for that site and worthy of administrative review. In addition, the operator can specify network ports that should not be accessed by external clients. Examples may include TCP ports 53 (DNS zone transfer), 143 (imapd), or 514 (syslogd). This category also includes the addition of a powerful feature for specifying a site surveillance policy to monitor user accesses to data and executable files (discussed in Section 3.2).

Other Security-relevant Events — provides other general security-relevant activity reports worthy of review by security administrators. This includes significant backward movement of the system clock beyond what is normally performed by clock synchronization protocols. This backward time movement is a possible indicator of an attacker attempting to manipulate file or log state to reduce the risk of detection. This category recognizes setuid enabling by non-administrative users, and suspicious symbolic link creation in publicly writable directories. Process execution by reserved accounts that are not intended to run applications (e.g., bin, sys) are also recognized. Finally, this category recognizes attempts to alter the underlying audit configuration, potentially in an effort to flood or starve *eXpert-BSM*.

3.2. File surveillance policy specification

eXpert-BSM provides a facility for specifying a surveillance policy over file reads, writes, and executions. Under this policy, the *eXpert-BSM* operator may specify groups of users, files, and directories, and then use these groups to specify surveillance policies regarding file accesses. This allows the operator to easily establish rules for generating immediate notification when users step outside their designated roles on the system.

For example, consider a consultant who is granted access to parts of a file server that also contains company sensitive data to which the consultant should not have access. The operator would, as a first line of defense, set access controls on files and directories to prevent the consultant's access to these sensitive file system areas. However, over time users who work in these areas may fail to continually manage the proper settings on these files and directories, or may create new sensitive files that by default allow the consultant access. The surveillance policy allows the operator to easily detect whenever the consultant accesses, or even attempts to access, files or directories in the sensitive areas of the file system.

There are three distinct components to be specified within an *eXpert-BSM* access policy specification. The first component, the *UserGroups* section, allows the operator to specify groups of users, which are then referenced in the access policy. The names specified under the user groups should be present as valid login names defined within the password file, and user names can appear in multiple lists.

The second section, *FileGroups*, allows the operator to specify a set of files and directories that may be referenced together as a group while enumerating the access policy. Files specified in the file groups should be fully qualified pathnames. The operator can also specify directories, as shown in the example surveillance policy specification in Figure 1. Files and directories can appear in multiple lists.

The third section is *Policy*, within which the operator can specify illegal read, write, and execute accesses between users and files. The policy section essentially defines access mode relations among user groups and file groups. For each user group entered in the policy, three possible relations can be specified: *nread*, *nwrite*, and *nexec*. The *nread* mode indicates that users in the associated list are not allowed to read files matching the file lists specified in the bracket clause. Illegal file writes and executions are specified similarly. Figure 1 presents an example of an *eXpert-BSM* access policy specification.

In Figure 1, there exists a small group of regular staff defined as *user1* and *user2*. There is a management staff, with one manager *admin* and an accounting group consisting of user *acct*. Four file groups are defined. The first file group is the programs group, where programs are defined as being located in */bin, /usr/bin, /usr/local/bin,* and */usr/local/ftp/bin*. An administrative tools group consists of files in */etc/bin, /etc/sbin, /usr/sbin,* and */sbin*. A directory containing company secrets is named */secret*. A payroll file group consists of a file called */accounting/DBMS/payroll.db*.

In Figure 1, regular staff are not allowed to read company secrets or payroll data, as specified by the associated *nread* function. Regular staff may not write to files in the company secrets, programs, payroll, or admin tools, and regular staff may not execute admin tools. If *eXpert-BSM* observes user activity that contradicts this policy, an alert is raised. Members of the management staff are not allowed to modify files in the program or admin tools file groups, but have

```
UserGroups { RegStaff   (user1 user2)
             Management (admin)
             Accounting (acct)
}
FileGroups { Programs ( /bin /usr/bin /usr/local/bin /usr/local/ftp/bin )
             Admtools ( /etc/bin /etc/sbin /usr/sbin /sbin )
             CompanySecrets  ( /secret )
             Payroll  ( /accounting/DBMS/payroll.db )
}
Policy {     RegStaff (
                   nread[CompanySecrets Payroll]
                   nwrite[CompanySecrets Programs Payroll Admtools]
                   nexec[Admtools] )
             Management(
                   nread[]
                   nwrite[Programs Admtools]
                   nexec[] )
             Accounting (
                   nwrite[Programs Admtools]
                   nread[CompanySecrets]
                   nexec[Admtools] )
}
```

Figure 1. Example surveillance policy.

unrestricted read and execute access over the entire system. Members of the accounting staff are not allowed to modify files in the program or admin file groups, read company secret files, or execute admin tools.

4. eXpert-BSM architecture and features

The preceding section discusses the threat coverage provided by the *eXpert-BSM* knowledge base. This section provides an overview of the features and capabilities of the *eXpert-BSM* distribution package, and its management when deployed on multiple hosts in a network.

4.1. Preprocessing the Solaris BSM event stream

eXpert-BSM runs on Solaris 2.6, 7, and 8, in 32-bit and 64-bit operating modes. These versions of Solaris have an auditing mechanism known as SunSHIELD Basic Security Module [21], or BSM for short. BSM has its roots in the C2 compatibility package for SunOS 4.x, developed to comply with the TCSEC [22] (Orange Book) requirements.

Before analyzing audit records, *eXpert-BSM*'s event preprocessing service, *ebsmgen*, first transforms the content of each audit record into an internal message structure. These messages include two important synthetic fields, called *synthetic_parentCmd* and *synthetic_parentIP*. Although audit records provide detailed information regarding each system call, they do not identify the command (process image name) under which the system call was invoked. The *synthetic_parentCmd* field tracks this important attribute by observing *exec* calls. Second, although Solaris audit records are structured to include information regarding source IP information for transactions not performed from the console, this information is unreliable across audit event types and OS versions. By tracking the source IP information and always reporting it in *synthetic_parentIP*, *ebsmgen* provides consistently correct IP information for all audit records.

Each message is passed on from the preprocessor to the event handling interface of the expert system, where it is asserted as a fact according to a fact type definition known as a *ptype* in P-BEST [10]. Figure 2 shows the ptype definition for audit records, as used in *eXpert-BSM*. Each field in the ptype corresponds to audit token data fields, except for the two synthetic tokens explained above.

Developing tools for analysis of BSM data is not without difficulty. The only tool provided with Solaris for audit data interpretation is *praudit*, which prints audit data in a simple text form. Although Solaris has some library routines for producing binary audit records, there are no routines available for consumers of BSM data. There is no formal grammar specified for BSM to help developers of consumer tools, and an effort to specify such a grammar for the BSM of Solaris 2.6 encountered some difficulties [6]. In Solaris 7 and 8, several new and initially undocumented audit token types were introduced. These are related to the support of 64-bit mode and other new features such as IPv6.

The syntax of audit records and audit tokens is relatively well specified in the documentation, but the semantics of the content is not. This is especially true for audit records

```
ptype[bsm_event
    human_time:          string,    ‘Header timestamp as a string.
    header_event_type:   int,       ‘Header event numerical ID
    header_time:         int,       ‘Header time as a numeric value.
    header_command:      string,    ‘Header event ID as a string (event name)
    header_size:         int,       ‘Header byte count
    msequenceNumber:     int,       ‘Sequence token number
    path_List:           string,    ‘Paths from one or several path tokens
    subject_auid:        int,       ‘Subject audit ID
    subject_euid:        int,       ‘Subject effective user ID
    subject_ruid:        int,       ‘Subject real user ID
    subject_pid:         int,       ‘Subject process ID
    subject_sid:         int,       ‘Subject audit session ID
    subject_machine_ID:  string,    ‘Subject machine ID
    in_addr_address:     string,    ‘In_addr Internet address
    in_addr_hostname:    string,    ‘In_addr Internet hostname
    attr_uidList:        int,       ‘Attribute owner UID
    val_List:            int,       ‘Argument value
    return_return_value: int,       ‘Return process value
    return_error_number: int,       ‘Return process error
    textList:            string,    ‘Text strings from one or several text tokens
    exec_args:           string,    ‘Exec arguments
    exec_env_txt:        string,    ‘Exec environment
    sock1_sock_type:     int,       ‘Socket type
    sock1_remote_port:   int,       ‘Socket remote port
    sock1_remote_iaddr:  string,    ‘Socket remote IP address
    sock1_local_port:    int,       ‘Socket local port
    sock1_local_iaddr:   string,    ‘Socket local IP address
    sock2_sock_type:     int,       ‘Socket type for second socket token
    sock2_remote_port:   int,       ‘Socket remote port for second socket token
    synthetic_parentCmd: string,    ‘Synthetic parent command
    synthetic_parentIP:  string     ‘Synthetic parent IP address
]
```

Figure 2. The P-BEST ptype for BSM events.

generated by applications outside the kernel, such as the *login* program. For developers of BSM audit trail analysis tools, such as the system described in this paper, this necessitates empirical studies of large amounts of audit data to understand the semantics of the BSM data stream. Undocumented changes between different versions of Solaris contribute to the difficulty of this task.

4.2. Modes of operation

eXpert-BSM requires no reactive probing of the system state, resulting in an IDS that produces identical results in batch and real-time modes. Batch-mode processing allows *eXpert-BSM* operators to process previously archived audit files, typically created by *auditd*. In real-time mode, *eXpert-BSM* is able to analyze audit records as they are produced by the kernel. The *eXpert-BSM* inference engine is packaged together with three additional modules that cooperate to relay BSM records directly from the kernel to the *eXpert-BSM* inference engine via interprocess communication, as illustrated in Figure 3. In its real-time operating mode, the *eXpert-BSM* package employs the following modules:

ebsmsetpolicy — (real-time mode) is a small setuid to root application that configures the desired audit policy for the kernel, then terminates immediately.

ebsmprobe — (real-time mode) establishes process-to-process communication between the Solaris kernel and *ebsmgen*. *ebsmprobe* runs setuid to root in order to read the audit records from the kernel.

ebsmgen — (batch and real-time modes) accepts and translates Solaris BSM audit records into EMERALD event messages as discussed in Section 4.1. An intermediate process is also used to manage buffers efficiently.

eXpert-BSM — (batch and real-time modes) is the EMERALD P-BEST-based forward-chaining expert system knowledge base and inference engine. It accepts event messages from *ebsmgen* and produces intrusion detection reports.

4.3. Alert message format

Within the EMERALD project, a format for alert messages has been developed, with both producer and consumer

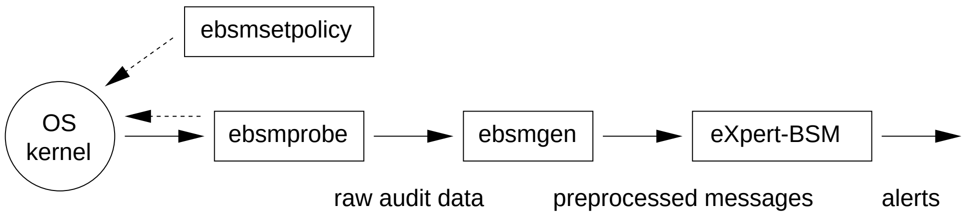

Figure 3. The components and data flow in the eXpert-BSM process chain (real-time mode).

processes in mind. Producers include monitors targeting diverse event streams, using different analysis techniques, such as the expert system used in *eXpert-BSM* or the probabilistic model of *eBayes* [23]. Typical consumers are alert management and presentation applications, correlation engines, and components handling automated attack countermeasures.

In Figure 4, an example alert from *eXpert-BSM* is shown. It reports that user *bob* successfully changed file permissions on a system executable file. Messages are encoded in a host-independent binary form suited for network transportation, but printed here in text form. Response recommendations are provided here both in a verbose format targeted for presentation to human operators and in a format aimed at automated response components.

4.4. Multi-host deployment

The host-based IDS concept is tightly coupled to the surveillance needs of individual host computing assets distributed throughout a network. This concept of distributed, lightweight sensors plays a central role in the EMERALD architecture and, consequently, the EMERALD infrastructure provides mechanisms for component configuration, message transmission, alert subscription, and distributed alert consolidation [13].

Currently, multi-host deployments of *eXpert-BSM* are supported through an alert collection application called *efunnel*, which multiplexes alert and status messages from several EMERALD monitors into a single message stream. Each *eXpert-BSM* is configurable to store its produced alarms locally on its host if desired, and can simultaneously forward these alerts to other subscribing security services located on remote systems distributed throughout the network. For example, *efunnel*, or its counterpart *eDBMS* for alarm storage into an SQL database, can be deployed and configured to subscribe to alert communications from a large suite of EMERALD monitors, including multiple *eXpert-BSM*s.

An example of multi-host deployment is illustrated in Figure 5. In this figure, EMERALD monitors are deployed across various key assets on a network. In addition to alarms, all EMERALD monitors (including *eXpert-BSM*)

```
Message ID 601 2001-02-03 01:23:19.761289 UTC:
 alert_report_ID = 3
 alert_thread_ID = 3
 alert_count = 1
 alert_gen_time = 2001-02-03 01:23:19.000000 UTC
 alert_start = 2001-02-03 01:23:19.000000 UTC
 alert_model_confidence = AL_CONFIDENCE_HIGH
 incident_class = CLASS_INTEGRITY_VIOLATION
 incident_signature = BAD_SYSTEM_BIN_MOD
 incident_description =
  Alteration to system executable
 observer_type = OB_TYPE_SIGNATURE
 observer_id = 102
 observer_stream = OB_STREAM_BSM
 observer_name = eXpert-BSM
 observer_version = 1.2
 observer_location = 192.168.2.20
 observer_src_file = realtime
 source_IParray = 192.168.1.100
 source_username = bob
 source_ruid = 0
 source_euid = 0
 source_auid = 2138
 source_pid = 6597
 target_IParray = 192.168.2.20
 outcome_generic = SUCCESS
 outcome_system_code = 0 (0x0)
 command = chmod(2)
 command_parent = /usr/bin/chmod
 resource_targetname = /usr/bin/gunzip
 resource_owner = root
 resource_owner_uid = 0
 recommendation =
   "Kill process 6597, Session ID 6542.
   Isolate and examine file [ /usr/bin/gunzip ].
   Lock out user account bob until you have
   determined who is responsible for this
   activity. Check the configuration parameter
   BSM_SYSTEM_BIN_LOCATIONS."
 recommendation_directives =
   "kill -pid 6542 -da 192.168.2.20
   lockout -uname bob -da 192.168.2.20
   fixperms -fn /usr/bin/gunzip -da 192.168.2.20
    -newattr 000
   checkcfg -da 192.168.2.20
    -name BSM_SYSTEM_BIN_LOCATIONS"
```

Figure 4. An example alert message from eXpert-BSM.

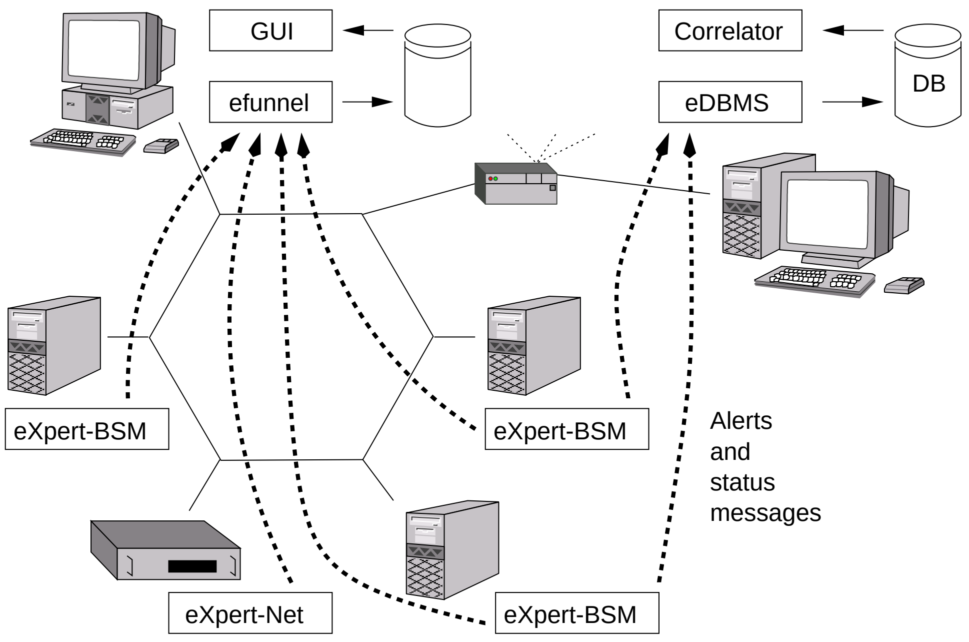

Figure 5. An example of multi-host deployment of eXpert-BSM.

are capable of forwarding health and status messages to their subscribers. Health and status messages can then be used to recognize unexpected shutdown or destruction of deployed EMERALD monitors, thus making it difficult to kill a monitor without detection by other INFOSEC services. In Figure 5, a suite of *eXpert-BSM* monitors forward their alerts to a host running *efunnel* operating on the same LAN. In addition, a subset of the *eXpert-BSM* monitors have a separate subscriber operating from a location external to the local network. The remote subscriber is managing an SQL database, and using *eDBMS* to store alert information from its selected sensors. On this remote data center, one may operate a correlation engine to examine the database of alert reports produced by selected monitors across this and other LANs.

5. Operational characteristics

eXpert-BSM has been packaged and distributed over the last four years to sponsors through private arrangement, and later as an evaluation release on the Internet to all who register. Throughout its development, *eXpert-BSM* has been subject to multiple third-party evaluations, batch testing, and red-team exercises as part of its participation in the DARPA Information Assurance Program. Experiences in operational deployments of *eXpert-BSM* have also greatly enhanced its capabilities, and has promoted attention to issues of optimization for real-world requirements. This section briefly summarizes activities that have enhanced the operational characteristics of *eXpert-BSM*.

5.1. Test battery

The *eXpert-BSM* distribution includes a test battery, containing a data collection that can be used in batch mode to trigger every intrusion model in the knowledge base. The user documentation contains both a description of the test data and a description of the expected output in terms of *eXpert-BSM* reports. In addition to being a basis for regression testing during development, this data set is useful to *eXpert-BSM* operators in several ways. Typically, an operator would like to be assured that the monitor works correctly after installation. When started in test mode, *eXpert-BSM* loads the configuration settings that are specific for the test battery, and starts a batch mode run that will exercise the analysis and presentation components of *eXpert-BSM*. This will show the operator all the possible types of reports that the monitor can produce. In addition, if the operator wants to run live attacks to make an end-to-end test in real time, the test battery documentation provides helpful instructions.

The test battery is designed to be a concentrated collection of attack data, with as little normal data as possible, to minimize the size of the data file and the time it takes to run through it. The purpose of the test is to validate the proper operation of all the intrusion models in *eXpert-BSM*, and to demonstrate the contents of the resulting reports. Although producing such data is time-consuming and sometimes difficult, we believe that this type of test battery is appreciated by IDS operators. Results from a questionnaire sent to registered *eXpert-BSM* operators strongly support this view.

Another form of testing is to run through large amounts of normal data to make sure that there are no false alarms. We continuously perform such testing of *eXpert-BSM* in our development and production environments. The DARPA IDS evaluation data from 1998 and 1999 is of this character [11], with several weeks of large data sets containing a small suite of attacks inserted for every day. The attacks selected are intended to exploit a strategically broad range of vulnerabilities that are representative of the major threats being used to infiltrate systems today.

5.2. Experimentation, deployment, and evaluation

Over the years of its development, versions of *eXpert-BSM* have been deployed in third-party laboratories, such as groups within the Air Force Research Laboratory and National Security Agency, for operational evaluations and experiments. In addition, these components have participated in multiple yearly live red-team exercises, mainly within the DARPA Information Assurance and Survivability suite of research programs. These activities have provided valuable input to the continuing development of the knowledge base and other features of *eXpert-BSM*.

In April 2000, the first release of *eXpert-BSM* was made available for download on the Internet. Those who registered their contact information were granted a time-limited evaluation license. At the time of writing, well over 200 organizations have registered.

We are currently aware of at least one military operational center and one commercial data center where the evaluation version has been fielded operationally to monitor critical servers. At both centers *eXpert-BSM* has been in continuous use for more than a year.

5.3. Optimization and Performance

In general, expert systems built with P-BEST are much faster than systems using the traditional interpreting model. This is because P-BEST code is translated to C, which is then compiled just like any C program [10]. In addition, P-BEST has undergone several modifications to further enhance its performance in terms of speed and integration with other programs. The modifications include language extensions that allow most C native types to be used in P-BEST, translator directives to pass some constructs directly to the C code, and an improved execution model for the inference engine. We have also developed C libraries that optimize the evaluation of complex antecedent expressions.

For any IDS analyzing a high-bandwidth event stream, it is important to be able to discard as much irrelevant data as possible as early in the process as possible. The *eXpert-BSM* knowledge base uses only 58 of the over 250 possible types of BSM audit records (auditable event types) in its intrusion models. In real-time mode, the Solaris audit kernel module is configured to produce only those 58 types of records. For batch mode, our preprocessing component *ebsmgen* performs the same selection. Our experiments show that for large sets of typical audit data ($>$ 1 GB), this preselection reduces the amount of data that need to be produced and processed to on average about 10% of the total amount that would be produced if full auditing were enabled.

The original auditd is designed to write audit records only to files. The *eXpert-BSM* package includes a component called *ebsmprobe* that replaces auditd, reads audit records directly from the kernel, and uses interprocess communication to pass the records to the preprocessing and analysis components for direct consumption. Thus, *eXpert-BSM* avoids expensive disk I/O operations for audit records and eliminates the need to reserve large amounts of disk space for audit files.

We recommend installing *eXpert-BSM* on local disk space rather than on network-mounted volumes, for better security and to avoid unpredictable file access delays. Internally, any kind of over-the-network access such as NIS or DNS lookup is avoided, except during the short initialization phase. Because many sites use NIS for user account information, *eXpert-BSM* uses its own local file for mapping numerical user IDs to usernames, which comprises the information in */etc/passwd* and NIS.

If the monitored host is running an extremely active process producing very large volumes of audit records, such as a heavily loaded DBMS, auditing can be turned off for that process to let the IDS be more responsive in its monitoring of the other processes on the host. We propose that a separate account be created for the sole purpose of running the heavy process, and that the account be excepted from auditing by an entry in */etc/security/audit_user*.

To obtain performance measurements, we have deployed *eXpert-BSM* on a Sun Enterprise 450, which is used as a file server and compute server for about 15 users. The machine is equipped with two UltraSparcII 400 Mhz processors, and 1 GB RAM. The additional load imposed by *eXpert-BSM* was studied in an experiment where we measured the completion time for building a relatively large software package, both in the presence and in the absence of the *eXpert-BSM* monitor. We ran *make* for a clean distribution of openssl-0.9.6 and measured the completion time as reported by */usr/bin/time*. A total of 10 runs were performed for each of the two situations, and each run was followed by other operations to eliminate the effects of file-system caches etc. When *eXpert-BSM* was not running, the 10 builds took on average 428 seconds each to complete, with a standard deviation of 0.8. With *eXpert-BSM* running in its “out-of-the-box” configuration, each build produced 94,684 audit event records, and took on average 454 seconds to complete, with a standard deviation of 1.1. We can conclude that the presence of the *eXpert-BSM* monitor caused a 6% increase in completion time for the task.

6. Related work

Operating system audit logs offer an interesting vantage point to the security-relevant operations of host systems. In [15], a design of effective auditing for security-critical systems is explored. Some standardization efforts for handling audit content have been examined [1], as have issues of what additional network related activity is worthy of representation in host audit trails [4]. A more recent work on applying formality to audit log structures is [6], which includes a discussion on some of the difficulties in automated BSM audit trail parsing.

There is a variety of related research efforts that explore what one can do with audit data to automatically detect threats to the host. An important work is MIDAS [20], as it was one of the original applications of expert systems—in fact using P-BEST—to the problem of monitoring user activity logs for misuse and anomalous user activity. CMDS, by SAIC, demonstrated another application of a forward-chaining expert-system, CLIPS, to a variety of operating system logs [18]. USTAT [9] offered another formulation of intrusion heuristics using state transition diagrams [16], but by design remained a classic forward-chaining expert system inference engine. ASAX [7] introduced the Rule-based Sequence Evaluation Language (RUSSEL) [12], which is tuned specifically for the analysis of host audit trails.

7. Conclusion

Host-based intrusion detection offers the ability to detect a wide variety of computer misuse through the direct analysis of process activity inside the host. Host-based analysis offers an important complement to network traffic analysis, providing threat detection coverage that is simply not easily available through the analysis of raw network traffic.

eXpert-BSM is a powerful and mature service for isolating security misuse and important security-relevant warning indicators. It analyzes the rich content of the Solaris BSM audit stream in real time, providing operators with distilled alert information and response recommendations. *eXpert-BSM* has been under development since 1998, and continues to progress in its effectiveness and usability through extensive testing, experimentation, and deployment experience.

eXpert-BSM is available for download at:

```
http://www.sdl.sri.com/emerald
```

Acknowledgments

The authors are grateful to all the people who have contributed to the design, implementation, evaluation and evolution of *eXpert-BSM* and other EMERALD components. We thank Martin Fong and Keith Skinner at SRI for their extensive contributions to the *eXpert-BSM* package. The feedback provided by a number of beta testers and users has been most helpful and is greatly appreciated. We also thank the OS security group at Sun Microsystems for answering our many questions about the Solaris audit subsystem.

References

[1] M. Bishop. A standard audit trail format. In *Proceedings of the 18th National Information Systems Security Conference*, pages 136–145, Baltimore, Maryland, Oct. 10–13, 1995. National Institute of Standards and Technology/National Computer Security Center.

[2] D. Bruschi, E. Rosti, and R. Banfi. A tool for pro-active defense against the buffer overrun attack. In J.-J. Quisquater et al., editors, *Computer Security – Proceedings of ESORICS 98*, volume 1485 of *LNCS*, pages 17–31, Louvain-la-Neuve, Belgium, Sept. 16–18, 1998. Springer-Verlag.

[3] CERT Coordination Center, Software Engineering Institute, Carnegie Mellon University, Pittsburgh, PA 15213-3890, USA. *Buffer Overflow in Sun Solstice AdminSuite Daemon sadmind*, Dec. 14, 1999. CERT Advisory CA-1999-16, *http://www.cert.org/advisories/CA-1999-16.html*.

[4] T. E. Daniels and E. H. Spafford. Identification of host audit data to detect attacks on low-level IP vulnerabilities. *Journal of Computer Security*, 7(1):3–35, 1999.

[5] J. de Haas. Vulnerability in Solaris ufsrestore. Bugtraq, June 14, 2000. *http://archives.neohapsis.com/archives/bugtraq/2000-06/0114.html*.

[6] C. Flack and M. J. Atallah. Better logging through formality: Applying formal specification techniques to improve audit logs and log consumers. In H. Debar, L. Mé, and S. F. Wu, editors, *Recent Advances in Intrusion Detection (RAID 2000)*, volume 1907 of *LNCS*, pages 1–16, Toulouse, France, Oct. 2–4, 2000. Springer-Verlag.

[7] J. Habra, B. Le Charlier, A. Mounji, and I. Mathieu. ASAX: Software architecture and rule-based language for universal audit trail analysis. In Y. Deswarte et al., editors, *Computer Security – Proceedings of ESORICS 92*, volume 648 of *LNCS*, pages 435–450, Toulouse, France, Nov. 23–25, 1992. Springer-Verlag.

[8] L. T. Heberlein et al. A network security monitor. In *Proceedings of the 1990 IEEE Symposium on Security and Privacy*, pages 296–304, Oakland, California, May 7–9, 1990.

[9] K. Ilgun. USTAT: A real-time intrusion detection system for UNIX. In *Proceedings of the 1993 IEEE Symposium on Security and Privacy*, pages 16–28, Oakland, California, May 24–26, 1993.

[10] U. Lindqvist and P. A. Porras. Detecting computer and network misuse through the production-based expert system

toolset (P-BEST). In *Proceedings of the 1999 IEEE Symposium on Security and Privacy*, pages 146–161, Oakland, California, May 9–12, 1999.

[11] R. Lippmann, J. W. Haines, D. J. Fried, J. Korba, and K. Das. Analysis and results of the 1999 DARPA off-line intrusion detection evaluation. In H. Debar, L. Mé, and S. F. Wu, editors, *Recent Advances in Intrusion Detection (RAID 2000)*, volume 1907 of *LNCS*, pages 162–182, Toulouse, France, Oct. 2–4, 2000. Springer-Verlag.

[12] A. Mounji. *Languages and Tools for Rule-Based Distributed Intrusion Detection*. PhD thesis, Institut d'Informatique, University of Namur, Belgium, Sept. 1997.

[13] P. G. Neumann and P. A. Porras. Experience with EMERALD to date. In *Proceedings of the 1st Workshop on Intrusion Detection and Network Monitoring*, Santa Clara, California, Apr. 9–12, 1999. The USENIX Association.

[14] A. One. Smashing the stack for fun and profit. *Phrack Magazine*, 7(49), Nov. 8, 1996. *http://www.fc.net/phrack/files/p49/p49-14*.

[15] J. Picciotto. The design of an effective auditing subsystem. In *Proceedings of the 1987 IEEE Symposium on Security and Privacy*, pages 13–22, Oakland, California, Apr. 27–29, 1987.

[16] P. A. Porras and R. A. Kemmerer. Penetration state transition analysis: A rule-based intrusion detection approach. In *Proceedings of the Eighth Annual Computer Security Applications Conference*, pages 220–229, San Antonio, Texas, Nov. 30–Dec. 4, 1992.

[17] P. A. Porras and P. G. Neumann. EMERALD: Event monitoring enabling responses to anomalous live disturbances. In *Proceedings of the 20th National Information Systems Security Conference*, pages 353–365, Baltimore, Maryland, Oct. 7–10, 1997. National Institute of Standards and Technology/National Computer Security Center.

[18] P. Proctor. Audit reduction and misuse detection in heterogeneous environments: Framework and application. In *Proceedings of the Tenth Annual Computer Security Applications Conference*, pages 117–125, Orlando, Florida, Dec. 5–9, 1994.

[19] T. H. Ptacek and T. N. Newsham. Insertion, evasion, and denial of service: Eluding network intrusion detection. Technical report, Secure Networks, Inc., Calgary, Alberta, Canada, Jan. 1998. *http://www.clark.net/˜roesch/idspaper.html*.

[20] M. M. Sebring, E. Shellhouse, M. E. Hanna, and R. A. Whitehurst. Expert systems in intrusion detection: A case study. In *Proceedings of the 11th National Computer Security Conference*, pages 74–81, Baltimore, Maryland, Oct. 17–20, 1988. National Institute of Standards and Technology/National Computer Security Center.

[21] Sun Microsystems, Inc., 901 San Antonio Road, Palo Alto, CA 94303, USA. *SunSHIELD Basic Security Module Guide, Solaris 7*, Oct. 1998. Part No. 805-2635-10.

[22] U.S. Department of Defense. *Trusted Computer System Evaluation Criteria*, Dec. 1985. DoD 5200.28-STD.

[23] A. Valdes and K. Skinner. Adaptive, model-based monitoring for cyber attack detection. In H. Debar, L. Mé, and S. F. Wu, editors, *Recent Advances in Intrusion Detection (RAID 2000)*, volume 1907 of *LNCS*, pages 80–92, Toulouse, France, Oct. 2–4, 2000. Springer-Verlag.